

FastPM: a new scheme for fast simulations of dark matter and halos

Yu Feng^{1*}, Man-Yat Chu¹, Uroš Seljak^{1,2}, Patrick McDonald²

¹*Berkeley Center for Cosmological Physics, Department of Physics, University of California Berkeley, Berkeley CA, 94720*

²*Lawrence Berkeley National Laboratory, Berkeley, CA, 94705*

July 7, 2016

ABSTRACT

We introduce FastPM, a highly-scalable approximated particle mesh N-body solver, which implements the particle mesh (PM) scheme enforcing correct linear displacement (1LPT) evolution via modified kick and drift factors. Employing a 2-dimensional domain decomposing scheme, FastPM scales extremely well with a very large number of CPUs. In contrast to COMoving-Lagrangian (COLA) approach, we do not require to split the force or track separately the 2LPT solution, reducing the code complexity and memory requirements. We compare FastPM with different number of steps (N_s) and force resolution factor (B) against 3 benchmarks: halo mass function from Friends of Friends halo finder, halo and dark matter power spectrum, and cross correlation coefficient (or stochasticity), relative to a high resolution TreePM simulation. We show that the modified time stepping scheme reduces the halo stochasticity when compared to COLA with the same number of steps and force resolution. While increasing N_s and B improves the transfer function and cross correlation coefficient, for many applications FastPM achieves sufficient accuracy at low N_s and B . For example, $N_s = 10$ and $B = 2$ simulation provides a substantial saving (a factor of 10) of computing time relative to $N_s = 40$, $B = 3$ simulation, yet the halo benchmarks are very similar at $z = 0$. We find that for abundance matched halos the stochasticity remains low even for $N_s = 5$. FastPM compares well against less expensive schemes, being only 7 (4) times more expensive than 2LPT initial condition generator for $N_s = 10$ ($N_s = 5$). Some of the applications where FastPM can be useful are generating a large number of mocks, producing non-linear statistics where one varies a large number of nuisance or cosmological parameters, or serving as part of an initial conditions solver.

1 INTRODUCTION

Extracting full information from observations of the large scale structure (LSS) of the universe, in weak lensing, galaxies, and other tracers, requires accurate predictions, which are only possible using simulations. These simulations can be used to create mock galaxy or weak lensing catalogs for co-variance estimation (Knebe et al. 2015), to vary the predictions as a function of cosmological or nuisance parameters (for example, galaxy formation parameters in halo occupation models as by Cooray & Sheth 2002), or even as a tool to generate initial conditions (Wang et al. 2013; Jasche & Wandelt 2013; Kitaura 2013). A full N-body simulation is too expensive both in terms of CPU-hours and wall-clock times. For this reason approximated N-body solvers that produces a reasonably accurate dark matter density field provides a practical alternative in critical applications. They range from simple Lagrangian perturbation theory field realizations (Kitaura, Yepes & Prada 2014; Monaco et al. 2013), to N-body simulations optimized for specific applications.

A large class of codes that employ the latter is the particle mesh (PM) family (e.g. Merz, Pen & Trac 2005; Carlson, White & Padmanabhan 2009; Heitmann et al.

2010; White et al. 2010; Tashev, Zaldarriaga & Eisenstein 2013; White, Tinker & McBride 2014). The idea is to ignore the force calculation on small scales by skipping the Tree (PP) force part of a full Tree-PM (P^3M) code. Vanilla PM has been criticized for failing to reproduce the linear theory growth at large scale as the number of time steps is reduced (see e.g. Izard, Crocce & Fosalba 2016). Recently, Tashev, Zaldarriaga & Eisenstein (2013) introduced the COMoving-Lagrangian enhanced particle mesh (COLA) scheme, where the large scale displacement is governed by the analytic calculation from second order Lagrangian theory (2LPT), and the particle mesh is used only to solve for the “residual” small scale displacement that affects the formation of halos. COLA scheme has gained a lot of attention recently (Tashev, Zaldarriaga & Eisenstein 2013; Tashev et al. 2015; Howlett, Manera & Percival 2015; Izard, Crocce & Fosalba 2016; Koda et al. 2015) because it adds a relatively small overhead to the vanilla particle mesh method, yet enforces the correct growth on large scales. As expected, this approach fails on small scales, where approximate tree solvers (Sunayama et al. 2015) proposed to reduce the frequency of updating the small scale force produce better results (percentage level agreement at high k), although at a higher compu-

tational cost. An alternative approach to reduce the cost of small scale interactions is to neglect the Tree force calculation for particles that have formed halos (Khandai & Bagla 2009).

The wall-clock time of a simulation is a somewhat overlooked issue (likely because the usual applications of approximate methods focus on a large number of independent mocks). The wall-clock time is still of relevance from a practical point of view, especially so for certain applications. For example, for Markov Chains Monte Carlo or similar sampling algorithms, a shorter wall-clock time in the simulations allows more steps per chain which can be of significant importance. There are two possible ways to reduce the wall-clock time. The first is to redraw the compromise between amount of calculation and accuracy – usually, less accurate results can be obtained by faster simulations. The second approach is to employ more computing resources. In the second approach the idea is that the implementation of the scheme can efficiently use a larger amount of computing resource – the so called strong scaling performance. Strong-scaling will become even more relevant as the number of computing nodes of super-computing systems grows with time. Usually, the more complex is the code, the harder it is to enforce strong scaling, specially when moving to new platforms.

To address some of these issues, we implement a simple PM scheme into a new code we call FastPM, where the linear theory growth of displacement (Zel’dovich approximation or 1LPT, where nLPT stands for Lagrangian Perturbation Theory at order n) is enforced by choosing an appropriate set of kick and drift factors. The particle mesh solver in FastPM is written from scratch to ensure that it scales extremely well with a large number of computing nodes. We implement it both within our approach, and within the COLA implementation, so that we can compare their performances.

We will describe the FastPM code in Section 2 of this paper. We discuss the numerical schemes briefly in Section 2.1, then move on to discuss the 2-dimensional parallel decomposition in Section 2.2 and show the strong scaling of FastPM in Section 2.3. In Section 3, we explore the parameter space (number of time steps and force resolution) and investigate favorable schemes for approximated halo formation.

2 THE FASTPM CODE

2.1 Numerical Schemes

The time integration in FastPM follows the Kick-Drift-Kick symplectic scheme described in Quinn et al. (1997), but is modified to achieve the correct large scale growth. This is discussed further below. Here we discuss the transfer functions for force calculation in Fourier space.

In a particle mesh solver, the gravitational force is calculated via Fourier transforms. First, the particles are painted to the density mesh, with a given a window function $W(r)$. We use a linear window of unity size (Hockney & Eastwood 1988, cloud-in-cell). We then apply a Fourier transform to obtain the over-density field δ_k . A force transfer function relates δ_k to the force field in Fourier space.

There are various ways to write down the force transfer function $\nabla\nabla^{-2}$ in a discrete Fourier-space. The

topic has been explored extensively by Hockney & Eastwood (1988) in the context of high resolution PM and PPPM simulations. FastPM supports several combinations, which we describe below.

- Naive: naive Green’s function kernels (k^{-2}) and differentiation kernels ($i\mathbf{k}$). This is used in COLA to solve for the residual motion with PM.

$$\nabla\nabla^{-2} = i\mathbf{k} k^{-2}. \quad (1)$$

- H-E: Sharpening Naive scheme by de-convolving the CIC mass assignment window. This is an extremely popular set of choice because it is shown to minimize the small scale error in force calculation,

$$\nabla\nabla^{-2} = i\mathbf{k} k^{-2} W_{\text{CIC}}^{-2}(k), \quad (2)$$

where $W_{\text{CIC}} = \Pi_{d=x,y,z} \text{sinc}^2 \frac{k_d L}{2N}$.

- Finite differentiation kernel (FastPM). The finite differentiation operator on a mesh is used, and no correction for mass assignment is applied. The 3-point second-order central differentiation operator in Fourier space is (also mentioned in Hockney & Eastwood 1988):

$$\nabla^{-2} = \left(\sum_{d=x,y,z} \left(x_0 \omega_d \text{sinc} \frac{\omega_d}{2} \right)^2 \right)^{-1}, \quad (3)$$

where $x_0 = L/N_g$ is the mesh size, and $\omega = kx_0$ is the circular frequency that goes from $(-\pi, \pi]$, and

$$\nabla = D_1(\omega) = \frac{1}{6} (8 \sin \omega - \sin 2\omega). \quad (4)$$

D_1 is derived in Hamming (1989). The D_1 filter is exactly the same as the 4-point discrete differentiation filter used in GADGET (Springel 2005), except we apply the function in Fourier space to simplify the implementation.

- Gadget: naive Green’s function kernels (k^{-2}), and with D_1 , applying sharpening corrections for mass assignment. This is the kernel used in Gadget for the PM part of the force.

$$\nabla\nabla^{-2} = iD_1(k) k^{-2} W_{\text{CIC}}^{-2}(k). \quad (5)$$

In solvers (e.g. TreePM) where the short range force is calculated separately, a Gaussian function $W_g(k) = \exp[-(kr_s)^2]$ is often added to apodize the long range force and suppress grid anisotropy, r_s being the smoothing scale of long range force (typical slightly larger than the force mesh cell sizes) (Bagla 2002; Springel 2005; Habib et al. 2013). It is unclear this is necessary for pure particle mesh simulations where there is no short range force: some authors have completely neglected the smoothing filter, yet still obtained reasonable results (Tassev, Zaldarriaga & Eisenstein 2013); authors in other fields have suggested that a high order Gaussian smoothing performs well for certain problems (Hou & Li 2007). We find that even a mild smoothing, with $r_s = L/N_m$ of the mesh resolution, significantly reduces the number of halos at the low mass end; we will discuss the effect due to choice of schemes in Appendix A. For consistency in the main text paper we use the FastPM finite differentiation scheme without smoothing, which as shown in Appendix A gives a sharper density field on larger scales, even though the kernels are less accurate on small scales in a traditional sense (Hockney & Eastwood 1988).

FastPM accepts an arbitrary list of time steps. Shortcuts for several commonly used time stepping schemes (including linear and logarithmic) are provided:

- Time steps that are linear in scaling factor a . Linear stepping improves halo mass function, but requires assistance for an accurate linear scale growth factor if number of time steps is small (Tassev, Zaldarriaga & Eisenstein 2013; Tassev et al. 2015; Howlett, Manera & Percival 2015; Izard, Crocce & Fosalba 2016).

- Time steps that are linear in $\log a$. Logarithmic stepping improves growth on large (linear) scales at the cost of underestimating small scales and the halo mass function (White, Tinker & McBride 2014).

- A hybrid scheme that controls the time step resolution in high redshift and low redshift independently:

$$(\delta a/a)^{-1} = \sqrt{(1/a_1)^2 + (a/a_2)^2}, \quad (6)$$

where a_1 controls the early time stepping and a_2 controls the late time stepping. An example value of $a_1 = 0.05$ and $a_2 = 0.025$ gives about 90 steps from $a = 0.025$ to $a = 1.0$ (Carlson, White & Padmanabhan 2009).

In this paper, we will focus on linear a stepping.

To compensate for the lack of short range resolution, the resolution of the force mesh can be boosted by a factor $B = N_m/N_g$. The size per side of the mesh (N_m) used for force calculation is B times the number of particles per side (N_g). Most recent authors of PM/COLA advocated a mesh of $B = 3$ in order to capture the non-linear formation of halos. (Tassev, Zaldarriaga & Eisenstein 2013; Izard, Crocce & Fosalba 2016), while others advocated for $B = 1$ (Howlett, Manera & Percival 2015). In FastPM B can be provided as a function of time. We will investigate the choice of B in later in this paper.

2.2 Memory Usage

FastPM implements both COLA and vanilla PM. COLA requires storing the two 2LPT displacement fields s_1 and s_2 , inducing a memory overhead relative to vanilla PM. The state vector of a single particle in FastPM contains the position, velocity and acceleration, in addition, the initial position is encoded into an integer for uniquely identifying the particles. When COLA is employed, we also store the s_1 and s_2 terms. Currently, the position of particles x is stored in double precision to reduce systematic evolution of round-off errors near the edge of the boxes, which can be concerning if stored in single precision. One can also store the relative displacement from the original position in a single precision, which can further reduce the memory consumption by a few percent.

The memory usage on the state vector is summarized in Table 1. The memory usage per particle is 56 bytes for vanilla PM and 80 bytes for COLA. To account for load imbalance and particles near domain surfaces (ghosts), we always over-allocate the memory storage for particles by a factor of $A > 1$. Therefore, the total memory usage for storing the state vector is $M_1 = 56AN_g^3$ for PM, and $M_1' = 80AN_g^3$ for COLA.

To avoid repeated conversion from particles to the mesh, FastPM creates two copies of the force mesh. The memory usage for the force mesh is $M_2 = 2 \times 4(BN_g)^3 = 8B^3N_g^3$. Note that the buffer for domain decomposition and for creation of a snapshot is overlapped with the force mesh, and it thus does not incur further memory allocation.

In summary, the total memory usage of FastPM is

$$M = M_1 + M_2 = \begin{cases} (56A + 8B^3)N_g^3 & \text{PM,} \\ (80A + 8B^3)N_g^3 & \text{COLA.} \end{cases} \quad (7)$$

Column	Data Type	Width (Bytes)
x	double	24
v	single	12
a	single	12
q	integer	8
s_1	single	12
s_2	single	12
Total PM		56
Total COLA		80

Table 1. Memory for the state vector, per particle.

A is typically bound by $1 < A < 2$. Therefore, the additional memory cost of COLA relative to vanilla PM to store s_1 and s_2 is: 37% \sim 40% for $B = 1$, 20% \sim 27% for $B = 2$, and 9% \sim 15% for $B = 3$.

2.3 Domain Decomposition

The domain decomposition in FastPM is 2-dimensional. Decomposing in 2 dimensions (resulting 1-dimension ‘pencils’ or ‘stencils’) is an effective way to deploy large Fourier transforms on a massively parallel scale (see e.g. Pippig 2013; Pekurovsky 2012). Some gravity solvers implement a new 2-dimensional Fourier transform library (e.g. Habib et al. 2013). We choose to reuse the publicly available implementation, PFFT by Pippig (2013) for its minimal design. We note that PFFT was also used to improve the scaling of P-GADGET by Feng et al. (2015).

FastPM decomposes the particles into the same 2-dimensional spatial domains of the real space Fourier transform mesh. In general, there are more mesh cells than number of particles (when $B > 1$), and it is therefore more efficient to create ghost particles on the boundary of a domain than creating ghost cells. We do not further divide the domain along the third dimension because that would induce further communication costs.

For large scale computations, a 2-dimensional decomposition has two advantages over 1-dimensional decomposition: 1) a smaller total surface area; and 2) a more balanced load. The surface area is directly proportional to the amount of communication for ghost particles. The surface area increases linearly with the number of processes ($O[P]$) for 1-dimensional decomposition; while for 2-dimensional decomposition, the scaling is close to $O[P^{1/2}]$. The unit of parallelism in 1-dimensional decomposition is a slab of $1 \times N_g \times N_g$. Therefore, when the number of processes is greater than the number of slabs ($P > N_g$), the Fourier transform becomes extremely imbalanced. The unit of parallelism in 2-dimensional decomposition is a pencil of $1 \times 1 \times N_g$, and the constraint is $P > N_g^2$. For a typical mesh size of 8,192 we used, the limit translates to $8,192^2 = 67,108,864$ processors for 2-d decomposition, a limit cannot be reached even with the next generation exa-scale facilities.

We point out that FastPM is not the first N-body solver implementing a 2-dimensional domain decomposing scheme. Previous implementations (e.g. Habib et al. 2013; Feng et al. 2015) mostly focused on weak-scaling of simulations to a large number of particles. The strong scaling of schemes that resolves galactic scale interaction (gravity, hydrodynamics, and feedback) typically suffer from the heavy imbalance as the average volume of a domain decreases, and requires over-decomposition of domains (Springel 2005; Menon et al. 2015). FastPM does

not usually suffer from the over-decomposition since it does not implement any small scale interaction, as we will discuss in the next section.

2.4 Strong Scaling of FastPM

We run a series of tests to demonstrate the strong scaling performance of FastPM on the Cray XC-40 system Cori at National Energy Research Supercomputing Center (NERSC). The test simulation employs 1024^3 (1 billion) particles in a box of $1024h^{-1}\text{Mpc}$ per side, and a resolution of $N_s = 40/B = 3$. The cosmology used is compatible to the WMAP 9 year data.

The minimal number of computing nodes to run the simulation (due to memory constraints) is 4, which translates to $N_p = 128$ computing cores. We then scale up the computing scale all the way up to $N_p = 8,192$ (a factor of 64) computing cores. We measure the time spent in force calculation, domain decomposition and the generation of the 2LPT initial condition. Time spent in file operations (IO) is not shown, since it follows closely to the status of the file system instead of the scaling of the code. We note however, that the IO backend of FastPM (bigfile) is capable of achieving the peak performance of the file system in the BlueTides simulation (Feng et al. 2015).

We report the results of the scaling tests in Figure 1. In the left panel, we see that the scaling of the total wall-clock time is close to the ideal $1/N_p$ law. The scaling of 2LPT initial condition hits a plateau when more than 4096 computing cores are used, but this is not of particular concern since the fraction to the total only amounts 3% percent even for the 8,192 core run.

The deviation from the $1/N_p$ law is shown in the right panel of Figure 1. We show the evolution of the total CPU-hours (cost) as we increase the number of cores. For ideal $1/N_p$ scaling, the line should be flat. We see that the total cost increases slowly with the number of cores, by a factor of less than 1.45 when the number of cores increased by a factor of 64. The increase of total cost is primarily due to the imbalance of particles that is associated with the decrease of volume per domain. For example, with the 8,192 core run, the most loaded domain handles 3 times of the average number of particles per domain. In a numerical scheme where small scale force is fully resolved (TreePM), the computing time increases very quickly with the growth of over-density, and this imbalance would have significantly increased the cost. However, in an approximated particle mesh solver (like FastPM), the dominating cost component is Fourier transform, the load of which is balanced relatively well thank to the 2-dimensional decomposition. The increase in synchronization time due to particle imbalance only slowly increase the total computing time. We point out that the relatively quick increase in cost at small number of cores ($N_p < 1,024$) is correlated with the crossing of communication boundaries of “local” / per-cabinet network on the Cray XC 40 system Cori where the test is performed.

OpenMP threading is usually invoked as a workaround for strong-scaling limitations (e.g. Feng et al. 2015). For FastPM this particular context is no longer relevant. In fact, running with multiple threads is always slower than running with equal number of processes, due to the lack of thread parallelism in the transpose phase

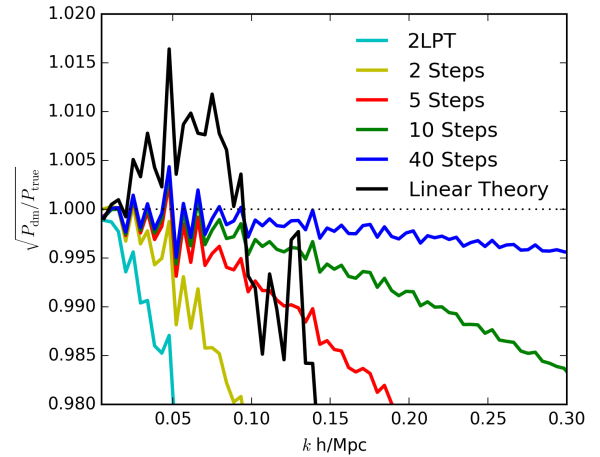


Figure 2. Recovery of linear growth with FastPM. We show the recovery rate $\epsilon(k)$ at quasi-linear scales, divided by full N-body (RunPB). Cyan: 2LPT; Yellow 2 steps; Red: 5 steps; Green: 10 steps; Blue: 40 steps. The fluctuations are due to the sampling variance errors in recovering the initial condition (see text). We also show the ratio of linear modes to full N-body, which show that linear theory deviates from the nonlinear results over most of the range.

of the parallel Fourier transform¹. Therefore on current computer architectures, we do not recommend using more than 1 OpenMP thread in FastPM.

2.5 Time Stepping

When the number of time steps is limited, the Kick-Drift-Kick integration scheme fails to produce the correct growth on large scales (e.g. Schneider et al. 2016; Izard, Crocce & Fosalba 2016). We note that even very large scales are not fully linear. As seen in figure 2, at $k_{th} \sim 0.03h/\text{Mpc}$, the linear theory model introduces an average 0.5% systematic error in mode amplitude at $z = 0$ due to nonlinear coupling of modes (see also Foreman, Perrier & Senatore 2015; Baldauf, Mercolli & Zaldarriaga 2015; Seljak & Vlah 2015; Takahashi et al. 2008). The linear scale growth error is especially severe with linear a time stepping, where the relative change in the growth factor can be large—for example, if the first two time steps are at from $a = 0.1$ ($z = 9$) to $a = 0.2$ ($z = 4$). Our numerical experiment shows that the error is already 1.5% with a single step from $a = 0.1$ to $a = 0.2$.

One way of eliminating this error without substantially increase the number of time steps is to insist the large scale growth follows a model. For example, the COmoving-Lagrangian (COLA) particle mesh scheme calculates the large scale trajectory of particles with second order Lagrangian Perturbation theory (Tassev, Zaldarriaga & Eisenstein 2013), which yields an accurate growth of the large scale modes. The disadvantages are higher memory requirement and the need to split the force into 2LPT and PM minus 2LPT, which requires tuning that is somewhat dependent on the number of time steps.

In a pure PM scheme the error is due to the incorrect assumption that force and velocity remains constant during the course of a time step. This assumption is violated

¹ Refer to <https://github.com/mpip/pfft/issues/6> for some discussions of this issue.

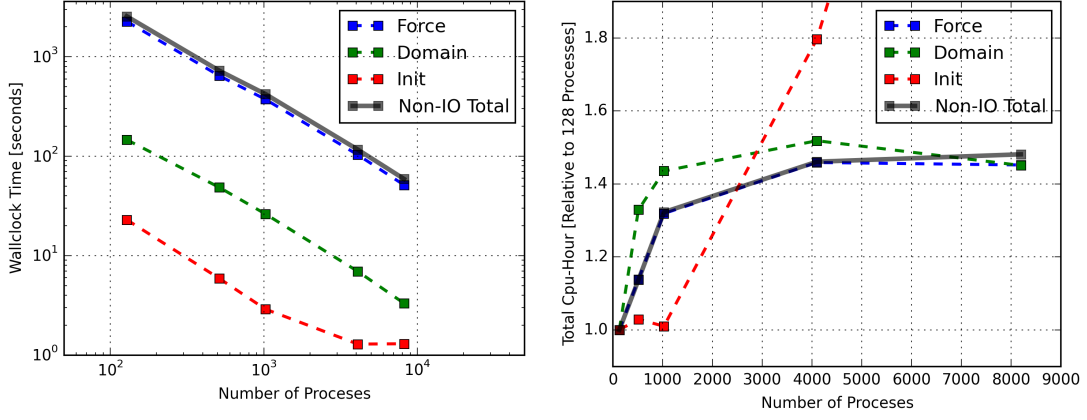


Figure 1. Strong scaling of FastPM. We perform the test on a $1024h^{-1}\text{Mpc}$ per side box on a 1024^3 particle grid. We run a total of 87 time steps (blendspace time stepping configuration) with the $B = 1, 2, 3$ force resolution scheme. We choose the variable force resolution scheme because in a traditional slab based particle mesh solver, the scaling would have stopped with 1024 cores for the $B = 1$ mesh. The three components shown are 1) Force: the time spent in obtaining acceleration of particles; 2) Domain: the time spent in migrating particles between different processors; 3) Init; the time spent in generating the 2LPT initial condition.

even in the linear regime. It can be compensated with a set of modified discrete drift and kick factors, motivated by the Zel'dovich (ZA) equation of motion. We derive next these factors (denoted with subscript FASTPM).

To see this, we first write down the Zel'dovich equation of motion to first order,

$$x_{\text{ZA}}(a) = q + D(a)s_1 \quad (8)$$

$$p_{\text{ZA}}(a) = a^2 \frac{dx_{\text{ZA}}}{dt} = a^2 \frac{dD}{da} \frac{da}{dt} s_1 \quad (9)$$

$$= \frac{dD}{da} a^3 E(a) s_1, \quad (10)$$

$$= a^3 E(a) g_p(a) s_1 \quad (11)$$

$$f_{\text{ZA}}(a) = a \frac{dp_{\text{ZA}}}{dt} = a^2 E(a) \frac{dp_{\text{ZA}}}{da} \quad (12)$$

$$= a^2 E(a) \frac{d[a^3 E g_p(a)]}{da} s_1 \quad (13)$$

$$= a^2 E(a) g_f(a) s_1. \quad (14)$$

We have followed the convention that $p = a^2 dx/dt$, $f = adp/dt$, and $E(a) = H(a)/H(a=1)$ is the dimensionless Hubble parameter. For simplicity we define the factors g_p and g_f as

$$g_p(a) = \frac{dD}{da} \quad (15)$$

$$g_f(a) = \frac{d(a^3 E g_p)}{da} \quad (16)$$

$$= \frac{d(a^3 E)}{da} \frac{dD}{da} + \frac{d^2 D}{da^2} a^3 E(a) \quad (17)$$

$$= 3a^2 E \frac{dD}{da} + a^3 \frac{dE}{da} \frac{dD}{da} + \frac{d^2 D}{da^2} a^3 E, \quad (18)$$

and their integrals as

$$G_p(a) = D(a) \quad (19)$$

$$G_f(a) = a^3 E(a) g_p(a) \quad (20)$$

Next, we rearrange the ZA equation of motion as drift/kick operators by integrating ZA over the time step and eliminating the ZA displacement s_1 from the equa-

tion of motion,

$$\Delta[x_{\text{ZA}}]_{a_0}^{a_1} = \Delta[D(a)]_{a_0}^{a_1} s_1 \quad (21)$$

$$= \frac{p_{\text{ZA}}(a_r)}{a_r^3 E(a_r)} \frac{\Delta[G_p]_{a_0}^{a_1}}{g_p(a_r)} \quad (22)$$

$$\Delta[p_{\text{ZA}}]_{a_0}^{a_1} = \frac{f_{\text{ZA}}(a_r)}{a_r^2 E(a_r)} \frac{\Delta[G_f]_{a_0}^{a_1}}{g_f(a_r)} \quad (23)$$

We can now define the FastPM modified drift and kick factor

$$\mathcal{D}_{\text{FASTPM}} = \frac{\Delta[x_{\text{ZA}}]_{a_0}^{a_1}}{p_{\text{ZA}}} = \frac{1}{a_r^3 E(a_r)} \left(\frac{\Delta[G_p]_{a_0}^{a_1}}{g_p(a_r)} \right) \quad (24)$$

$$\mathcal{K}_{\text{FASTPM}} = \frac{\Delta[p_{\text{ZA}}]_{a_0}^{a_1}}{f_{\text{ZA}}} = \frac{1}{a_r^2 E(a_r)} \left(\frac{\Delta[G_f]_{a_0}^{a_1}}{g_f(a_r)} \right). \quad (25)$$

These operators can be used to construct any finite integration steps, which integrate the equation of motion of ZA solution exactly. Note that we keep a_r as the reference time for the step. In a standard Kick-Drift-Kick scheme, a_r is a_0 at the first step, but otherwise a_r is between a_0 and a_1 .

It is trivial to show these factors converges to the usual drift and kick operators when the time steps are small. The usual drift and kick operators are (Quinn et al. 1997)

$$x(a_1) = x(a_0) + p(a_r) \int_{a_0}^{a_1} \frac{1}{a^3 E} da \quad (26)$$

$$p(a_1) = p(a_0) + f(a_r) \int_{a_0}^{a_1} \frac{1}{a^2 E} da, \quad (27)$$

of which, the Drift and Kick factors are

$$\mathcal{D}_{\text{PM}} = \int_{a_0}^{a_1} \frac{1}{a^3 E} da \quad (28)$$

$$\mathcal{K}_{\text{PM}} = \int_{a_0}^{a_1} \frac{1}{a^2 E} da. \quad (29)$$

When $a_1 \rightarrow a_0$, by definition we have

$$\frac{\Delta G_p}{\Delta a} \rightarrow g_p, \frac{\Delta G_f}{\Delta a} \rightarrow g_f, \quad (30)$$

and

$$\mathcal{D}_{\text{FASTPM}} \rightarrow \mathcal{D}_{\text{PM}}, \mathcal{K}_{\text{FASTPM}} \rightarrow \mathcal{K}_{\text{PM}}. \quad (31)$$

Therefore, on infinitesimal time steps, the two sets of operators are identical.

In Figure 2, we show the large scale power spectrum of FastPM divided by the full N-body simulation. We also show the comparison to the linear theory prediction, which deviates from nonlinear already at very low k . Even with 2 time steps, i.e. a single full time step beyond 2LPT, the modified scheme is able to match the nonlinear growth of a full N-body simulation at $k \sim 0.02h/\text{Mpc}$, a significant improvement over 2LPT or linear theory. In contrast, full N-body codes often do not match each other at very low k (Heitmann et al. 2008; Schneider et al. 2016), with constant offsets between them, likely due to inaccuracies in the linear growth. We thus believe our modified Kick-Drift scheme could be useful even for standard high resolution simulations.

We point out that it is possible (as we have done in the first version of this paper and FastPM) to calibrate the Kick or Drift factor against linear theory, or nonlinear power spectrum measured from a low resolution PM simulation in the limit of many steps, which gives nearly identical results. Both time stepping schemes are also implemented in FastPM.

3 DARK MATTER AND HALO BENCHMARKS WITH FASTPM

In this section, we investigate the accuracy of FastPM against computational cost, varying several of its parameters. We focus particularly on halo formation with FastPM, but we also compare it against the dark matter statistics. The particular application we have in mind is to find an approximate halo formation scheme that can be used for a fast extraction of galaxy statistics. To be more specific, we would like to find an approximation scheme that reduces the cost (CPU time), while also reducing the systematic error of the halo statistics to an acceptable level (defined more precisely below).

The parameters we investigate are number of time steps N_s and the force resolution $B = N_m/N_g$, the ratio between force mesh and number of particles per side. We also perform a comparison between (our version of) COLA and FastPM. The FastPM simulations used in this work are listed in Table 2.

The simulations are compared against a TreePM simulation on 2048^3 particles in a $1380h^{-1}\text{Mpc}$ per side box (RunPB). For the FastPM simulations, we reconstruct the s_1 and s_2 2LPT terms from a single precision $z = 75$ initial condition of RunPB, and extrapolate the initial displacement to $z = 9.0$. This ensures that the FastPM simulations are using the same initial modes as RunPB, up to the numerical errors. In all simulations, the Friend-of-Friend finder uses a linking length of 0.2 times the mean separation of particles (Davis et al. 1985). Our results are robust against linking length, as the main purpose of linking length is to identify over density features and apply abundance matching. In Appendix C, we also compare the results of $N_s = 10/B = 2$ simulations against those from a shorter linking length of 0.168, which reduces the halo bridging effect of FoF.

We list the total CPU time used in each run in Figure 3. The total CPU time increases with B and N_s . The most expensive scheme we considered is the $N_s = 40/B = 3$ scheme suggested by Izard, Crocce & Fosalba (2016), using 1300 CPU hours each on the reference system (70 times of 2LPT). Reducing the number of time steps and the force resolution can drastically reduce the computing

Model	Number of Steps	Force Resolution
COLA	5	3
PM	5	3
COLA	10	3
PM	10	3
COLA	10	2
PM	10	2
COLA	20	3
PM	20	3
COLA	40	3
PM	40	3

Table 2. List of Simulations.

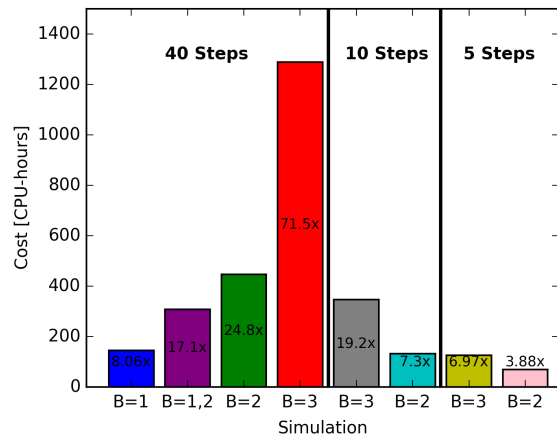


Figure 3. The integrated CPU-hours. The text in each vertical bar shows the ratio to generating a 2LPT initial condition. COLA and PM use almost same amount of CPU-hours, thus are not independently shown.

time. For example, while $N_s = 40/B = 3$ costs 72 times 2LPT, our favorite scheme $N_s = 10/B = 2$ uses only 7 times of cost of 2LPT, while $N_s = 5/B = 2$ reduces this to 4.

3.1 Definitions of benchmarks

We use the ratio of mass function $\phi_1(M)/\phi_2(M)$ to illustrate the difference in mass function.² We use abundance matching to correct for the difference in mass function, but note that more complicated alternatives have been used by other authors as well (e.g. Sunayama et al. 2015; Izard, Crocce & Fosalba 2016).

We define the transfer function T as the square root of the ratio of the power spectra

$$T(k, \mu) = \sqrt{P_1(k, \mu)/P_2(k, \mu)}. \quad (32)$$

The agreement is defined as good when T is close to 1. The transfer function measures the relative bias between two fields.

Note that transfer function is often defined as the ratio of the cross-power spectrum to auto-power spectrum: the two definitions are the same if the cross correlation

² 1 stands for the approximated model and 2 stands for the accurate model.

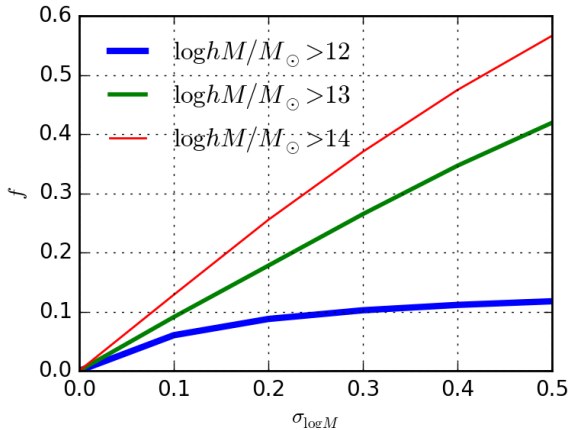


Figure 4. Stochasticity induced by mass scattering. We measure the stochasticity (large scale) due to adding mass scattering (given by σ) to the halo catalog. Three different abundance cut, corresponds to $M = 10^{12,13,14} h^{-1} M_{\odot}$ are shown in colors. The stochasticity increases with the mass scattering, which we identify as the main source of scale independent stochasticity. (See text)

coefficient is unity. We define the cross correlation coefficient r as

$$r(k, \mu) = P_{1,2}(k, \mu) / \sqrt{P_1(k, \mu) P_2(k, \mu)}, \quad (33)$$

where $P_{1,2}$ is the cross power between the approximated and the accurate model. For halos, we always use the catalog after abundance matching them. We do this by rank ordering them by assigned halo mass, and then selecting the same number of the most massive halos for the two catalogs.

From this we can define another related quantity, the dimensionless stochasticity,

$$f(k, \mu) = \sqrt{(1 - n P_1(k, \mu))(1 - n P_2(k, \mu))} + (1 - n P_{1,2}(k, \mu)), \quad (34)$$

where n is the number density of halos, which is identical in two simulations due to the abundance matching. The stochasticity is 1 when two catalogs contains completely different halos, and 0 when two catalogs are identical. So stochasticity f expresses the fraction of misidentified halos in the approximate simulation. Typically this happens because the code has assigned an incorrect mass to the halo, so that the halo does not enter the abundance matched catalog. We could have also defined stochasticity using a mass error instead, but we do not pursue this here. Our definition of stochasticity is k dependent: if the halo positions are incorrect then we expect stochasticity f to increase with k , before decreasing again to converge to the shot noise limit.

While the numerical inaccuracies are one source of stochasticity, in practice we also have observational limitations. We typically observe galaxy luminosity, and select all the galaxies above a certain luminosity threshold (which can change with redshift). But there is a scatter between luminosity and halo mass and as a consequence a galaxy catalog does not correspond to the most massive halos at the same abundance (we are ignoring satellite galaxies in this discussion). To investigate this we take a given halo mass catalog, add scatter to it, and rerank the halos. We then determine stochasticity, i.e. the fraction of halos that are the same between this catalog and the original one, at the same abundance.

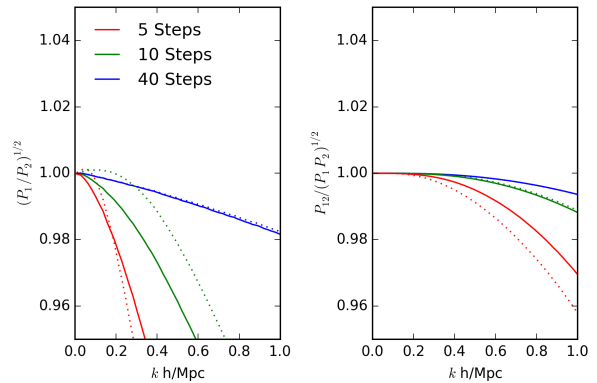


Figure 5. Benchmarks on matter density, varying number of time steps. Left: transfer function. Right: cross correlation coefficient. Colors: $N_s = 5$ (red), 10 (green), and 40 (blue) and 200 (cyan). Solid : PM. Dots : COLA. 200 Step run is with a lower resolution ($B = 2$). The 200 step (cyan) and 40 step (blue) line has overlapped in the right panel.

In Figure 4 we show the increase of stochasticity induced by increasing the mass scatter (in $\log M$) in the RunPB halo catalog. For example, the scatter of 0.23, as expected in BOSS CMASS catalog, corresponds to a stochasticity of 20% for typical CMASS halo mass (Rodríguez-Torres et al. 2015). So as long as FastPM stochasticity is significantly smaller than this there is no need to make it exactly zero.

It is worth pointing out that stochasticity or cross-correlation coefficient is in some sense the more important benchmark than the transfer function. This is because if the correlation coefficient is unity (or stochasticity zero) one can still recover the true simulation by multiplying the modes of the approximate simulation by the transfer function. Of course this requires the transfer function to be known, but it is possible that it is a simple function of parameters, such that one can extract it from a small set of simulations without a major computational cost. In the following we will however explore all of the benchmarks defined here. These are summarized in table 3 for each simulation.

3.2 Varying Number of Time Steps

In this section, we discuss the effects due to varying the number of time steps employed in the simulation N_s , while fixing the force resolution at $B = 3$.

The transfer function and cross correlation coefficient of matter density relative to the RunPB simulation is shown Figure 5. With 5 steps, the cross correlation coefficient is 93% at $k = 1 h/\text{Mpc}$. Increasing the number of time steps does improve transfer functions and cross correlation coefficients significantly. With 40 steps, the transfer function is close to 98% at $k = 1 h/\text{Mpc}$, and the cross correlation coefficient is close to 99% at $k = 1 h/\text{Mpc}$. These metrics indicate that if a high accuracy matter density field is of interest a 40 step simulation is preferred. It is worth pointing out that by extracting the transfer function, and then multiplying the modes with it, one obtains nearly perfect results up to $k = 1 h/\text{Mpc}$ even with 10 steps, given that the cross correlation coefficient is 99% or larger over this range.

We also observe that with the $N_s = 10$ runs COLA gives a larger transfer function than FastPM at all scales,

Samples	Transfer Function [†]	Cross Correlation Coefficient [†]	Stochasticity [†]	Mass Function
Matter	Y	Y	N	N/A
$\log h^{-1} M/M_{\odot} \geq 12$	Y	Y	Y	Y
$\log h^{-1} M/M_{\odot} \geq 13$	Y	Y	Y	Y
$\log h^{-1} M/M_{\odot} \geq 14$	Y	Y	Y	Y

[†] For these benchmarks, The halo mass in FastPM simulations are reassigned by abundance matching against halos in RunPB.

Table 3. List of Benchmarks.

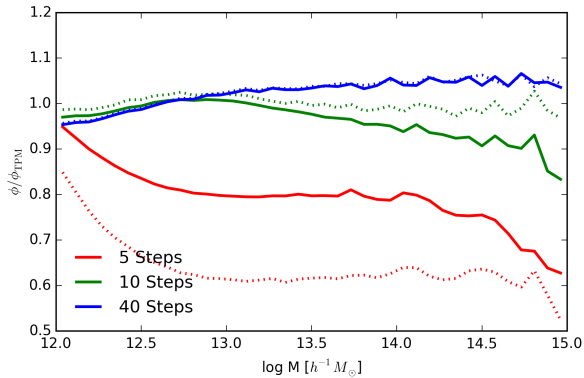


Figure 6. Benchmark of halo mass function, varying number of time steps. We show the number of times steps (color: red, green, blue) from $N_s = 5, 10$ and 40 . Solid: PM. Dots: COLA.

while $N_s = 5$ COLA gives a smaller transfer function and a significantly worse cross correlation coefficients at all scales. It is worth noting that COLA has a free parameter n_{LPT} which needs to be tweaked for number of steps and cosmology parameters (Tassev, Zaldarriaga & Eisenstein 2013). In our tests we find that COLA is very sensitive to the choice of this parameter. In contrast, the kick and drift scheme in FastPM does not require tweaking of any parameters.

We next look at the halos. Before applying abundance matching, we first show the mass function relative to RunPB is shown in Figure 6. For runs with more than 10 steps, the mass function has converged to 10% agreement with RunPB regardless of whether COLA or PM is used. However, if mass accuracy is required then 5 steps appears to be insufficient, as the 5 step PM run recovers only 80% of the mass function (60% for COLA). This is not necessarily a problem, since exact mass assignment is not required in a galaxy survey with a given abundance, as long as the rank ordering is preserved. However, in practice lower number of steps also introduces errors in the mass assignment, which increase the stochasticity, as we show below.

The rest of the benchmarks are calculated after applying abundance matching to reassign halo masses in the FastPM simulations. In Figure 7, we show the benchmarks on the transfer function, cross correlation coefficient and stochasticity of halos as the total number of time steps N_s is varied. Three mass threshold, $M = 10^{(12,13,14)} h^{-1} M_{\odot}$ are used. All the results are for $z = 0$. We find the following results:

1) Beyond 10 steps the improvement is very limited. This is very different from the matter density field, where one gains major advantage using 40 steps. The additional steps therefore mostly improves the profile and velocity dispersion of halos.

2) For abundance matched halos, PM out-performs COLA at low number of time steps. This can be most clearly seen in the 5 step runs, where PM is nearly a factor of 2 closer to exact solution at all k, μ values. The PM advantage over COLA decreases as the number of time steps increases. At 40 steps, PM and COLA converges to the same result. COLA splits the displacement into a residual field and a large scale 2LPT component. It is worth noting that even though the 10 step COLA simulations give better matter transfer function than PM (as seen in Figure 5), the advantage in matter density seem to be consistently hurting the performance in halos. This could be because COLA has free parameters whose performance may have been optimized for the dark matter benchmarks. In contrast, there are no free parameters in FastPM.

3) FastPM matches more massive halos better than less massive halos. For example, for the 10 step runs, the stochasticity reduces from 10% at $M > 10^{12} h^{-1} M_{\odot}$ to 7% at $M > 10^{14} h^{-1} M_{\odot}$. However, the overall stochasticity is given by f/n , and since n is a lot higher for the low mass halos the absolute stochasticity is a lot lower for low mass halos, despite the larger value of f . A 10% stochasticity is likely more accurate than our current level understanding of the halo mass - galaxy luminosity relation (Behroozi, Conroy & Wechsler 2010; More et al. 2009; Yang, Mo & van den Bosch 2009). For example, the scatter in the halo mass at a fixed luminosity is 0.4 dex for $10^{12} h^{-1} M_{\odot}$ halos (Behroozi, Conroy & Wechsler 2010), if all uncertainties are considered, and even larger for the larger halo masses. As a comparison, for a scatter of 0.18 dex in halo mass, we introduce a stochasticity of $f \sim 0.10, 0.18, 0.22$ for halos of mass $10^{12,13,14} h^{-1} M_{\odot}$. Hence we believe that the stochasticity levels generated by $N_s = 10$, or even $N_s = 5$, suffice given the current observational uncertainties in halo mass determination.

4) Redshift space statistics ($\mu > 0$, where $\mu = k_{\parallel}/k$ and k_{\parallel} is the component of the wavevector k along the line of sight) are not very different from the real space statistics ($\mu = 0$). Hence redshift space distortions do not significantly affect the conclusions above.

5) Scale dependence of f is weak for low mass halos, gradually increasing to higher masses. For the highest mass bin we observe it to increase by 0.05 to $k = 1 \text{ h/Mpc}$. We expect that a low resolution PM gets the halo centers wrong by a fraction of their virial radius: hence, for lower mass halos the absolute error is smaller.

Based on these observations one does not need more than 10 steps if halos at $z = 0$ are of interest, and indeed for many applications even a 5 step FastPM, possibly corrected with the transfer function, suffices. In addition, we comment that the requirement on the number of steps are similar at higher redshifts (we tested up to $z = 1$). Since our steps are uniform in expansion factor a , a 10 time step FastPM simulation that runs to $z = 0$ would naively

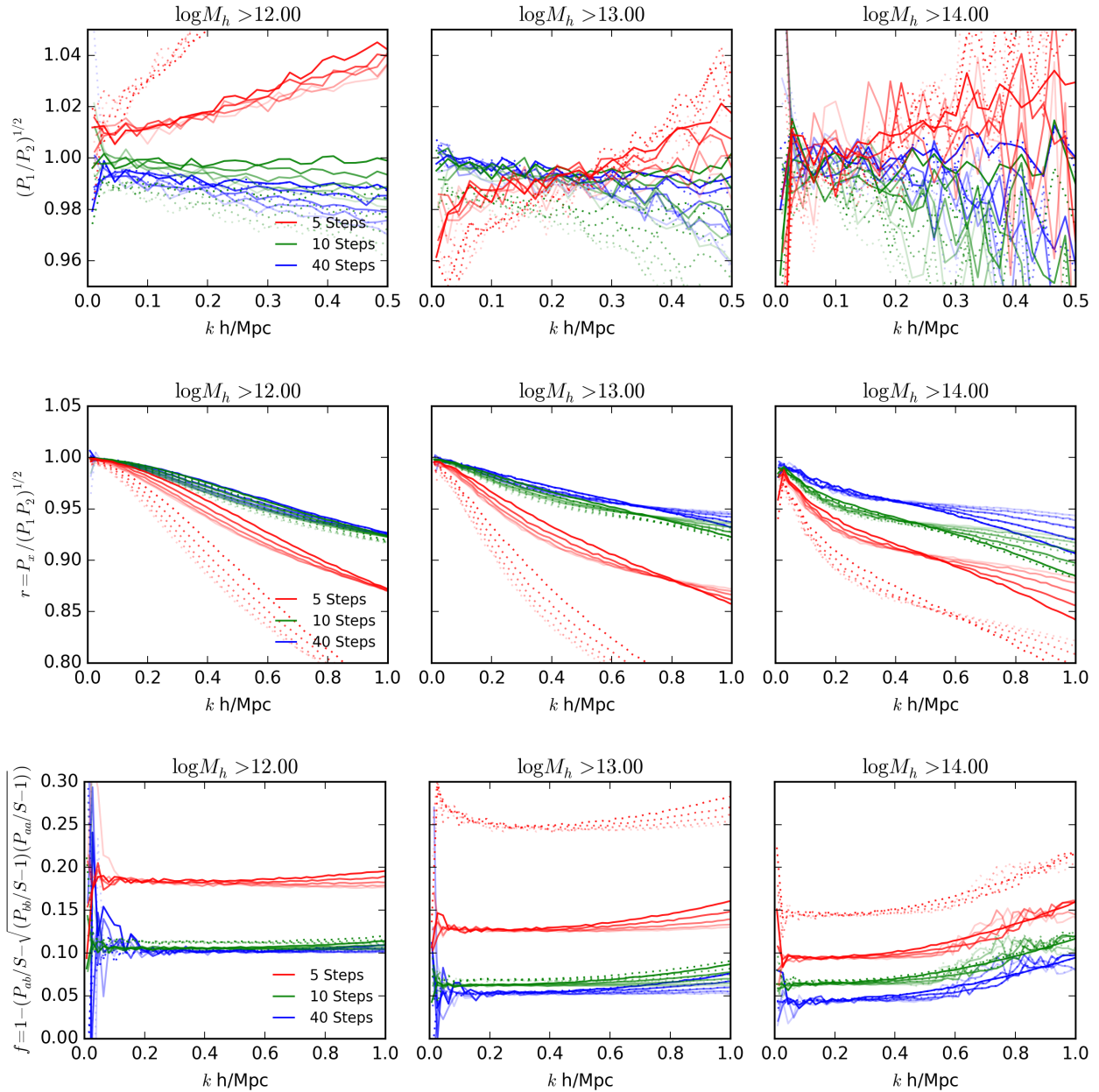


Figure 7. Benchmarks on halos, varying number of time steps. Top panel: transfer function. Center panel: cross correlation coefficient. Bottom panel: stochasticity. Colors: $N_s = 5$ (red), 10 (green), and 40 (blue). Opacity: Line of sight angle, $\mu = 0.1$ (transparent), 0.3, 0.5, 0.7, 0.9 (opaque). Solid : PM. Dots : COLA. $S = 1/n$ is the shot-noise level.

have the effective performance of 5 steps at $z = 1$, but we observe that actual performance is more in between 5 and 10 steps.

3.3 Varying Force Resolution B

In this section, we discuss the effects due to varying the resolution of the force mesh employed in the simulation B . As we have shown $N_s = 10$ is of sufficient accuracy for halos, so we will fix the number of time steps at $N_s = 10$ in this section.

The transfer function and cross correlation coefficient of dark matter density relative to the RunPB simulation is shown Figure 8. We see that going from $B = 3$ to $B = 1$ the transfer function at $k < 1h/\text{Mpc}$ barely changes, and the cross correlation coefficient changes even less.

As in the previous section, before applying abun-

dance matching we investigate the mass function benchmark in Figure 9. We see that regardless of whether PM or COLA is employed, as long as $B \geq 2$, the mass function is recovered at 90% level for $M > 10^{12} h^{-1} M_\odot$. However, with a lower resolution, $B = 1$, only 80% of the halo mass function is recovered at $M = 10^{13} h^{-1} M_\odot$, and even fewer halos are found at lower masses. This indicates that due to the low resolution, the density contrast in $B = 1$ is insufficient for detecting halos of $M \leq 10^{13} h^{-1} M_\odot$, as also seen in Lukić et al. (2007). However, one may still be able to salvage information about halos in the regime where Friend-of-Friend finder fails. For example, it is possible to combine a stochastic sampling method (e.g. QPM or PATCHY White, Tinker & McBride 2014; Kitaura, Yepes & Prada 2014) for less massive halos, but this may increase stochasticity and move f closer to unity.

The rest of the benchmarks are calculated after ap-

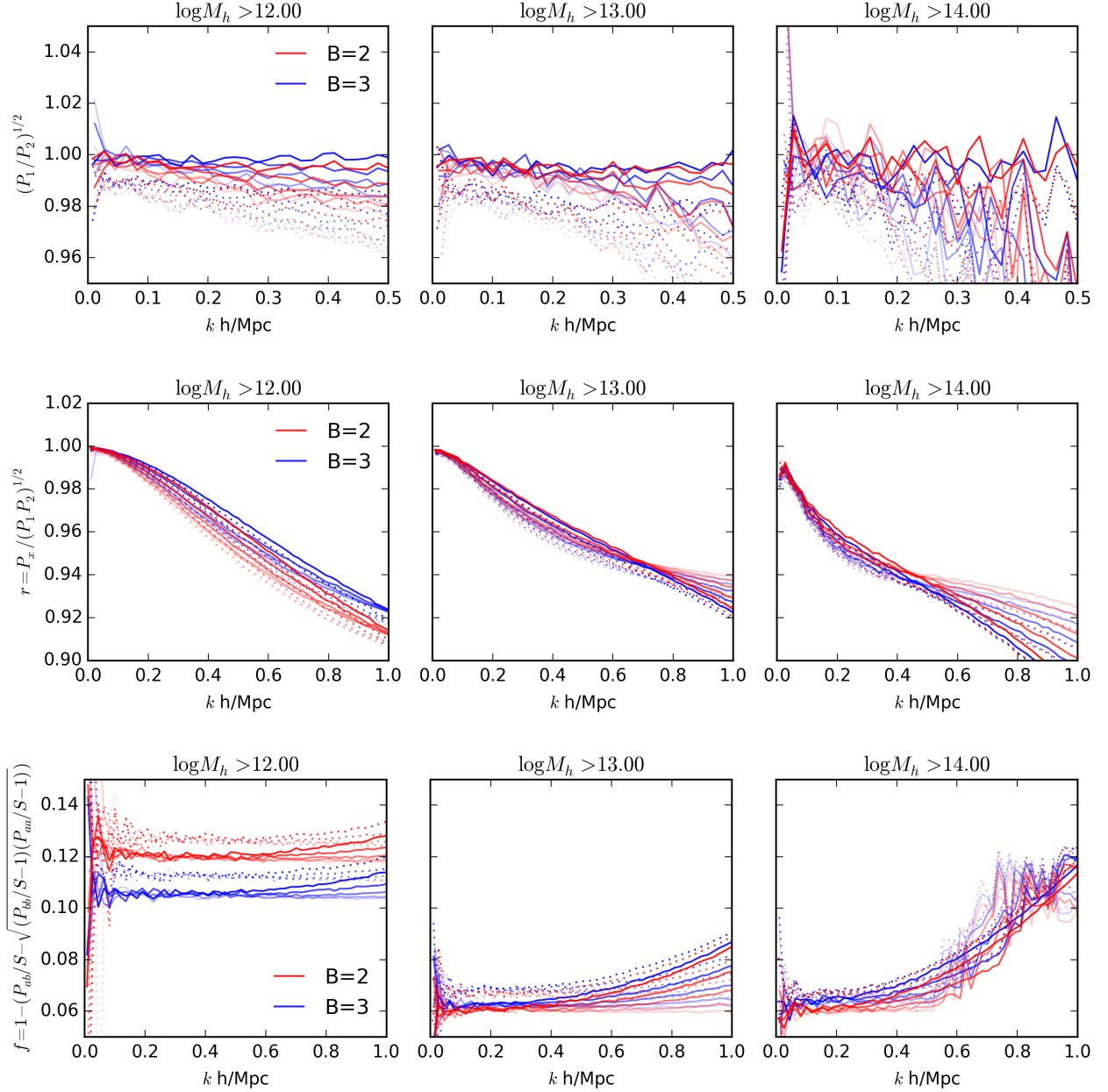


Figure 10. Benchmarks on halos, varying force resolution. Top panel: transfer function. Center panel: cross correlation coefficient. Bottom panel: stochasticity. Red: $B = 2$; Blue: $B = 3$; Solid: PM; Dots: COLA. Opacity: Line of sight direction, $\mu = 0.1$ (transparent), 0.3, 0.5, 0.7, 0.9 (opaque). Solid : PM. Dashed : COLA. $S = 1/n$ is the shot-noise level.

plying abundance matching to reassign halo masses in the FastPM simulations. We show the rest of the benchmark suite in Figure 10, which have been calculated after abundance matching. The structure of the figure is similar to Figure 7, but now we vary the force resolution. We again observe some slight advantages of PM comparing to COLA (one percent level). The improvement due to increasing the force resolution from $B = 2$ to $B = 3$ is very limited, typically at 1% level. The $B = 2$ approximation has a slightly larger (by 2%) stochasticity than $B = 3$ approximation for the least massive threshold ($M > 10^{12} h^{-1} M_{\odot}$). Given that $B = 2$ simulation is almost 3 times faster than a $B = 3$ simulation (Figure 3), the 2% increase in stochasticity is a reasonable price to pay. Overall, we find that the $N_s = 10/B = 2$ approximation uses about 10% of CPU time of a $N_s = 40/B = 3$

simulation, yet the benchmarks on halos of both are almost identical.

4 CONCLUSIONS

In this paper we introduce FastPM, a new implementation of an approximate particle mesh N-body solver. FastPM modifies the standard kick and drift factors such that it agrees with Zeldovich solution on large scales, guaranteeing zero error in $k \rightarrow 0$ limit even for very few time steps. These modified factors assume acceleration during the time step is consistent with 1LPT, and they reduce to the standard ones in the limit of short time steps. We recommend the use of these modified factors whenever correct large scale evolution is desired.

The domain decomposition in FastPM for parallel

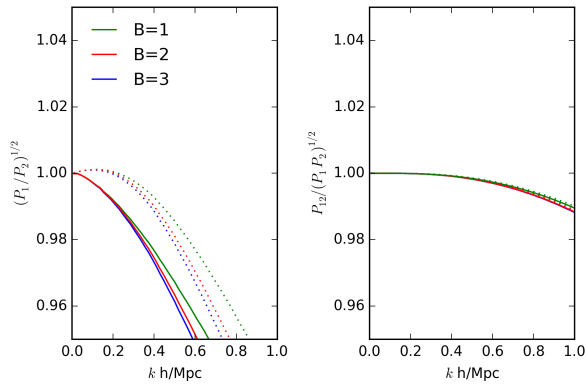


Figure 8. Benchmarks on matter density, varying force resolution. Red: $B = 2$; Blue: $B = 3$; Gray: $B = 1$. Solid: PM; Dots: COLA. The number of steps is fixed to 10.

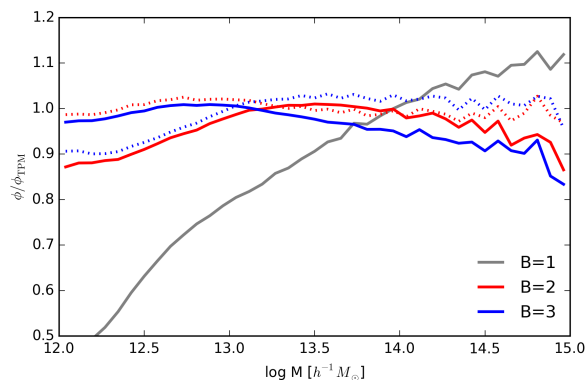


Figure 9. Benchmark on halo mass function, varying force resolution. Green: $B = 1$; Red: $B = 2$; Blue: $B = 3$. Solid: PM; Dots: COLA. The number of steps is fixed to 10.

Fourier Transform and particle data is in 2-dimensions, allowing the code to scale almost linearly with the number of CPUs when a large number of CPUs (more than 10,000) are employed.

We then proceed to investigate numerical precision of halos identified with FoF within FastPM. Four benchmarks are defined, measuring the performance: ratio of halo mass function, transfer function, cross correlation coefficient, and stochasticity. We show that

- our implementation of PM (with modified kick and drift factors) performs slightly better than COLA on all benchmarks; especially when the number of time steps is low (5 steps).
- the benchmarks on halos of any scheme with $N_s \geq 10/B \geq 2$ is very close to the exact solution. This makes the $N_s = 10/B = 2$ approximation very interesting, as it uses only 7 times of the computing time of generating a 2LPT initial condition, and 10% of time of a $N_s = 40/B = 3$ approximated run. For higher redshifts, or for cases where large stochasticity can be tolerated, the $N_s = 5/B = 2$ can also be adequate, at only 4 times the computing time of a 2LPT initial condition.

We see several use cases for FastPM:

- As the halo catalog step of a mock factory, FastPM can be useful for generating a large number of mock catalogs. This is the same use case scenario similar to other recently proposed approximate N-body codes

(Izard, Crocce & Fosalba 2016; Howlett, Manera & Percival 2015; Sunayama et al. 2015).

- Non-linear power spectrum (and higher order statistics) emulator: compared to other codes FastPM can efficiently utilize a very large amount of computing resources. In fact, the turn around time with FastPM can be as low as 1 minute if sufficient computing resources are reserved for real time usage. This means a large number of models, varying both cosmological parameters and nuisance parameters (such as halo occupation distribution parameters), can be explored rapidly. We plan to implement and expose a programming interface for various cosmology models in FastPM.

- Initial conditions solver: FastPM is designed as a software library, making it easy to embed into another application. One option is the initial conditions solver (Wang et al. (2013)), which requires derivatives as a function of initial modes. For the derivatives to be computationally feasible one needs a simple force evaluation scheme and very few time steps, both satisfied by FastPM. The scheme we have proposed, $B = 2$, $N_s = 10$, uses 7 times more CPU time than a single 2LPT step, but provides realistic friend-of-friend halos. Depending of the allowed stochasticity budget, using $N_s = 5$ or lower may also prove useful for computing the derivatives, given that the complexity of derivatives scales with N_s .

- While we have focused on halos in this paper, FastPM can also be used for other applications, such as weak lensing (where dark matter is used). When it comes to dark matter our recommended strategy is to multiply the density field with the transfer function, which can be obtained from FastPM itself ran at a higher resolution and number of time steps, or from a higher resolution simulation. Since the cross-correlation coefficient is very close to 1 up to $k = 1h/\text{Mpc}$, this would therefore guarantee high precision at least up to that k value. We expect the transfer function to be a slowly varying function of cosmological parameters and redshift. One possible strategy is to calibrate FastPM against at sufficient number of points in parameter space to make this error negligible. Finally, we expect the loss of precision at even higher k to be related to the structure of halos at small scales: FastPM already finds all the halos with the correct central position, and their mass error is relatively small, so only the internal halo profiles are incorrect. Since these are affected by baryonic effects (gas profile, AGN feedback etc.) anyways this needs to be addressed in any pure dark matter code, and low resolution FastPM is not necessarily limited relative to a high resolution N-body code. Another potential application is Lyman alpha forest, where a nonlinear transformation of matter density can be used as an approximation to the Lyman-alpha flux, with all three applications above being of possible interest. We plan to present some of these applications in the future.

Acknowledgment

We acknowledge support of NASA grant NNX15AL17G. The majority of the computing resources are provided at NERSC through the allocations for the Baryon Oscillation Spectroscopic Survey (BOSS) program and for the Berkeley Institute for Data Science (BIDS) program. We thank Dr. Jun Koda for distributing the source code of `cola_halo` under the GPLv3 license, which served both as a design inspiration and as a reference implementation of the COLA scheme. We thank Alejandro Cervantes and Marcel Schmittfull for their generous help in test-

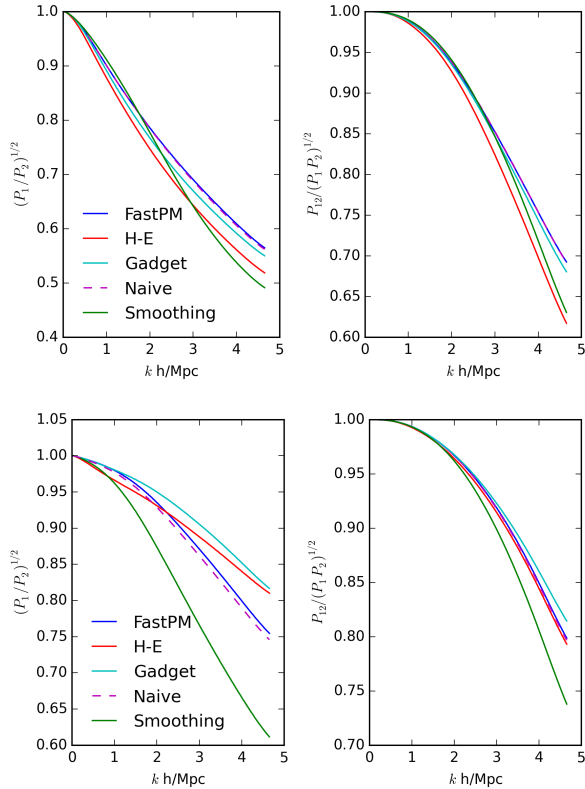


Figure A1. Comparing $z = 0$ power spectrum resulted from different schemes on the same initial condition with 10 and 40 step simulations. Upper panels : 10 steps; Lower panels: 40 steps. The reference is the RunPB simulation. The configuration of the simulations is the same as in Section 3, starting from $z = 9$.

ing the code. We thank Martin White for providing the RunPB TreePM simulation and initial condition that formed the foundations of our benchmarks. The development of FastPM is hosted by [github.com](https://github.com/rainwoodman/fastpm) at <https://github.com/rainwoodman/fastpm>. The version of code used in this paper is based on commit 9219d0. The data analysis software `nbodykit` is used for identifying Friend-of-Friend halos and calculating of power spectra.³ We welcome collaboration on development of both software packages.

APPENDIX A: CHOICE OF DIFFERENTIATION SCHEME

FastPM supports a variety of differentiation schemes. (Section 2). We explore the effect on dark matter density field due to choice of scheme in this section. This effect is shown in Figure A1, where we compare the $z = 0$ power spectrum resulted from different schemes on the same initial condition with 10 step and 40 step simulations against the RunPB simulation. The configuration of the simulations is the same as in Section 3, with a force mesh resolution of $L/N_m = 0.34h^{-1}\text{Mpc}$. Various schemes show overall a good level of agreement at $k < 0.5h/\text{Mpc}$. The difference increases to 5 percent at $k = 2h/\text{Mpc}$. At 10 steps, the FastPM kernel and the Naive kernel produces

³ <https://github.com/bccp/nbodykit>, separate publication being prepared.

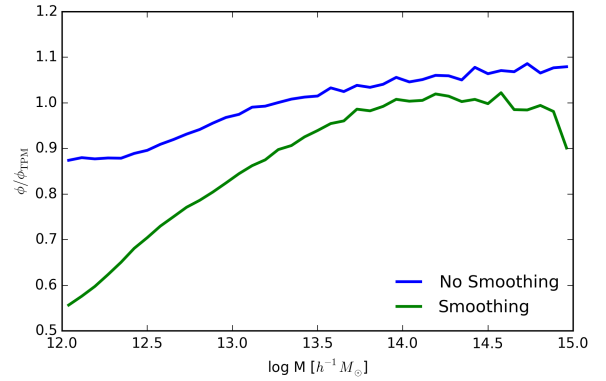


Figure A2. Comparing $z = 0$ mass function with and without smoothing on the same initial condition with 10 step simulations. The reference is the RunPB simulation. Blue: FastPM scheme without smoothing; Green: With smoothing. The configuration of the simulations is the same as Section 3.

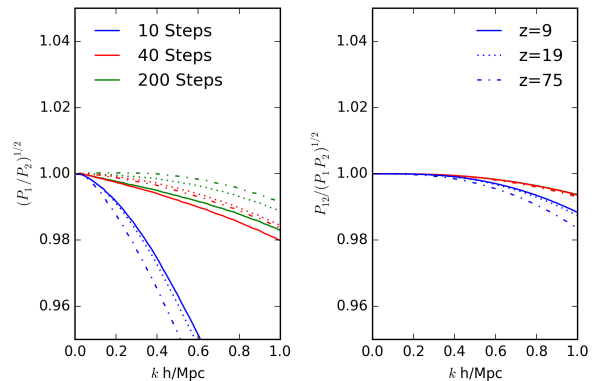


Figure B1. Comparing $z = 0$ power spectrum resulted from different starting redshifts on the same initial condition with 10 (blue), 40 (red), and 200 (green) step simulations. We show three starting redshifts, $z_i = 9$ (solid), 19 (dotted), 75 (dotted-dashed). The reference is the RunPB simulation, which started at redshift $z_i = 75$. Left: the square root of the ratio of power spectrum. Right: the cross correlation coefficient. Note that in the right panel the 200 step lines are covered by the 40 step lines.

the largest power and the highest cross correction coefficient. At 40 steps, the Gadget kernel shows improvements over FastPM and Naive kernel at $k > 1h/\text{Mpc}$. Smoothing (Gaussian with $r_s = L/N_m$) slightly increases the cross correlation coefficient at the cost of severely suppressing the power on small scales that corresponds to the collapse of halos, losing halos below $M < 10^{13.5}h^{-1}M_\odot$ (See Figure A2).

APPENDIX B: CHOICE OF STARTING REDSHIFT

Starting redshift affects the accuracy of simulations by two competing effects.

1) The temporal resolution decreases as the starting redshift increases. This effect is the most evident in the 10 step simulations, where the time steps are coarse. We see that the $z_i = 75$ simulation performed worse than $z_i = 9$ and $z_i = 19$. Therefore, for simulations with very few time steps, a lower starting redshift improves the benchmarks.

2) The stochasticity in initial condition decreases as the starting redshift increases. In our case we want to minimize it against that of the reference simulation. We see this with the 200 steps simulation, where the $z_i = 75$ simulation, which is the starting redshift of reference TreePM simulation, has the least error in power spectrum. The improvement due to better agreement in initial condition only affects the power spectrum, not the cross-correlation coefficient. Therefore, for simulations with many time steps, matching the starting redshift to that of the reference N-body simulation gives the best benchmarks. This is not surprising because the large scale gravity force in the reference N-body simulation is calculated with the same Particle Mesh method, although with many more time steps. The effect between $z_i = 19$ and $z_i = 75$ is about 0.4% in the power spectrum at $k = 1h/\text{Mpc}$ at $z = 0$.

APPENDIX C: CHOICE OF LINKING LENGTH

The friends-of-friend algorithm that we use to identify objects are known to bridge halos into the same object. The effect can affect our halo detection because in FastPM the density field has less contrast than an accurate N-body simulation. Reducing linking length to 0.168 can alleviate this problem and improve the quality of the mock halo catalog (Tinker et al. 2008). In Figure C1 we show that reducing linking length from 0.2 to 0.168 does not affect the benchmarks, except for the least massive bin ($10^{12}h^{-1}M_\odot$), where we observe a constant 2% increase in bias and stochasticity.

References

- Bagla J. S., 2002, *Journal of Astrophysics and Astronomy*, 23, 185
- Baldauf T., Mercolli L., Zaldarriaga M., 2015, *Phys. Rev. D*, 92, 123007
- Behroozi P. S., Conroy C., Wechsler R. H., 2010, *ApJ*, 717, 379
- Carlson J., White M., Padmanabhan N., 2009, *Phys. Rev. D*, 80, 043531
- Cooray A., Sheth R., 2002, *Phys. Rep.*, 372, 1
- Davis M., Efstathiou G., Frenk C. S., White S. D. M., 1985, *ApJ*, 292, 371
- Feng Y., Di-Matteo T., Croft R. A., Bird S., Battaglia N., Wilkins S., 2015, *ArXiv e-prints*
- Foreman S., Perrier H., Senatore L., 2015, *ArXiv e-prints*
- Habib S., Morozov V., Frontiere N., Finkel H., Pope A., Heitmann K., 2013, in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '13*, ACM, New York, NY, USA, pp. 6:1–6:10
- Hamming R. W., 1989, *Digital Filters* (3rd Ed.). Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK
- Heitmann K. et al., 2008, *Computational Science and Discovery*, 1, 015003
- Heitmann K., White M., Wagner C., Habib S., Higdon D., 2010, *ApJ*, 715, 104
- Hockney R. W., Eastwood J. W., 1988, *Computer Simulation Using Particles*. Taylor & Francis, Inc., Bristol, PA, USA
- Hou T. Y., Li R., 2007, *Journal of Computational Physics*, 226, 379
- Howlett C., Manera M., Percival W. J., 2015, *Astronomy and Computing*, 12, 109
- Izard A., Crocce M., Fosalba P., 2016, *MNRAS*, 459, 2327
- Jasche J., Wandelt B. D., 2013, *MNRAS*, 432, 894
- Khandai N., Bagla J. S., 2009, *Research in Astronomy and Astrophysics*, 9, 861
- Kitaura F.-S., 2013, *MNRAS*, 429, L84
- Kitaura F.-S., Yepes G., Prada F., 2014, *MNRAS*, 439, L21
- Knebe A. et al., 2015, *MNRAS*, 451, 4029
- Koda J., Blake C., Beutler F., Kazin E., Marin F., 2015, *ArXiv e-prints*
- Lukić Z., Heitmann K., Habib S., Bashinsky S., Ricker P. M., 2007, *ApJ*, 671, 1160
- Menon H., Wesolowski L., Zheng G., Jetley P., Kale L., Quinn T., Governato F., 2015, *Computational Astrophysics and Cosmology*, 2, 1
- Merz H., Pen U.-L., Trac H., 2005, *New A*, 10, 393
- Monaco P., Sefusatti E., Borgani S., Crocce M., Fosalba P., Sheth R. K., Theuns T., 2013, *MNRAS*, 433, 2389
- More S., van den Bosch F. C., Cacciato M., Mo H. J., Yang X., Li R., 2009, *MNRAS*, 392, 801
- Pekurovsky D., 2012, *SIAM Journal on Scientific Computing*, 34, C192
- Pippig M., 2013, *SIAM J. Sci. Comput.*, 35, C213
- Quinn T., Katz N., Stadel J., Lake G., 1997, *ArXiv Astrophysics e-prints*
- Rodríguez-Torres S. A. et al., 2015, *ArXiv e-prints*
- Schneider A. et al., 2016, *J. Cosmology Astropart. Phys.*, 4, 047
- Seljak U., Vlah Z., 2015, *Phys. Rev. D*, 91, 123516
- Springel V., 2005, *MNRAS*, 364, 1105
- Sunayama T., Padmanabhan N., Heitmann K., Habib S., Rangel E., 2015, *ArXiv e-prints*
- Takahashi R. et al., 2008, *MNRAS*, 389, 1675
- Tassev S., Eisenstein D. J., Wandelt B. D., Zaldarriaga M., 2015, *ArXiv e-prints*
- Tassev S., Zaldarriaga M., Eisenstein D. J., 2013, *J. Cosmology Astropart. Phys.*, 6, 36
- Tinker J., Kravtsov A. V., Klypin A., Abazajian K., Warren M., Yepes G., Gottlöber S., Holz D. E., 2008, *ApJ*, 688, 709
- Wang H., Mo H. J., Yang X., van den Bosch F. C., 2013, *ApJ*, 772, 63
- White M., Pope A., Carlson J., Heitmann K., Habib S., Fasel P., Daniel D., Lukic Z., 2010, *ApJ*, 713, 383
- White M., Tinker J. L., McBride C. K., 2014, *MNRAS*, 437, 2594
- Yang X., Mo H. J., van den Bosch F. C., 2009, *ApJ*, 695, 900

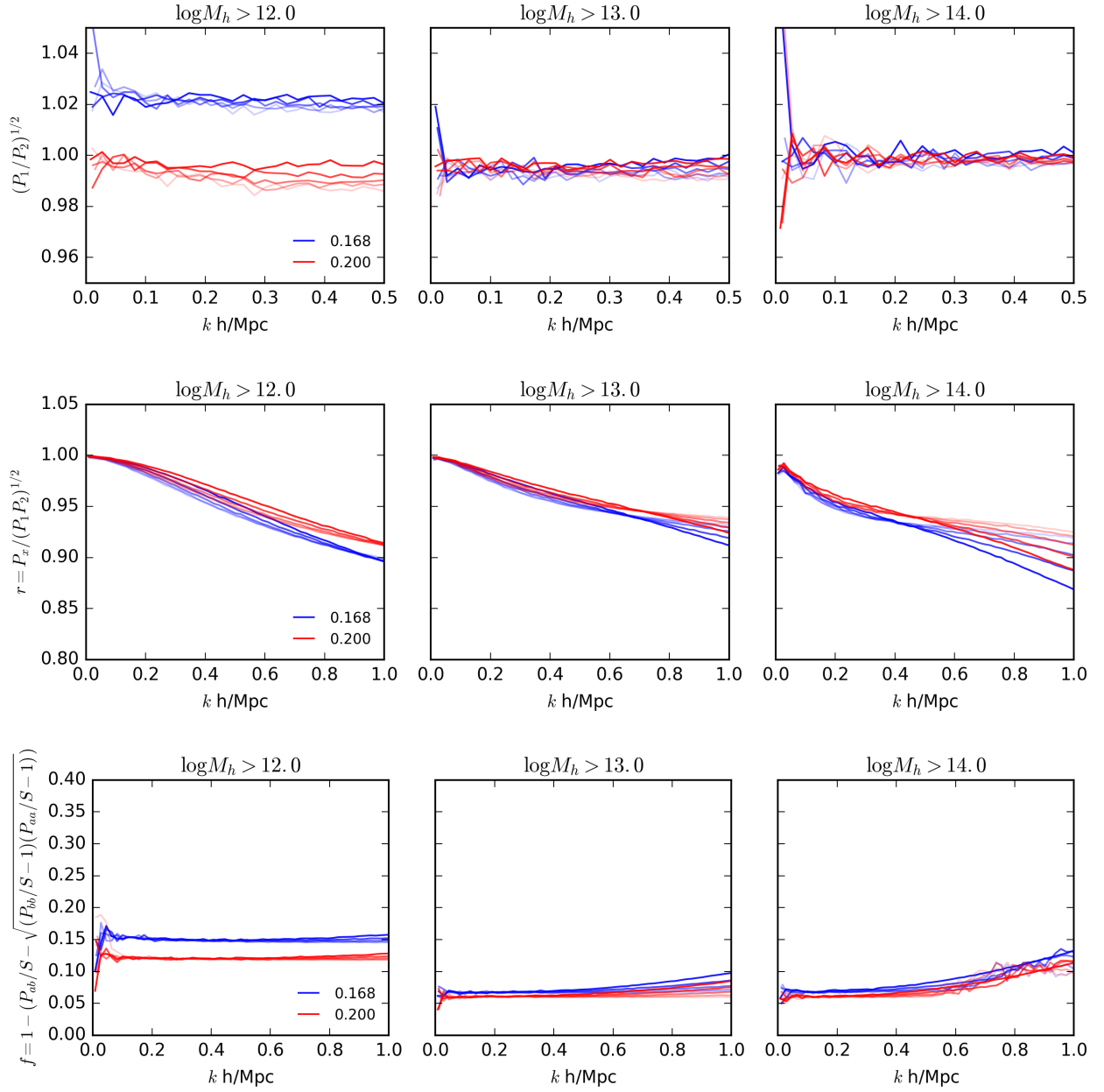


Figure C1. Benchmarks on halos, varying linking length. Top panel: transfer function. Center panel: cross correlation coefficient. Bottom panel: stochasticity. Red: 0.200 ; Blue: 0.168; Opacity: Line of sight direction, $\mu = 0.1$ (transparent), 0.3, 0.5, 0.7, 0.9 (opaque). $S = 1/n$ is the shot-noise level.